PCA

```
###-------------------------------------------------------------------------------------
###-------------------------------------------------------------------------------------
### PCA
###-------------------------------------------------------------------------------------
###-------------------------------------------------------------------------------------
library(mvtnorm)
library(ellipse)

# -------------------------------------------------------------------------------------
# 1. Read the data
#    ATTENZIONE alle etichette!
#    Se ci sono etichette (labels):
#    (dopo il "read.table"):    labels = data[, c(1:2)]
#                               data = data[,-c(1:2)]
# -------------------------------------------------------------------------------------
#       | X1 | X2 | .. | Xp |
#   -------------------------
#   | 1 | .. | .. | .. | .. |
#   | 2 | .. | .. | .. | .. |
#   |   | .. | .. | .. | .. |
#   | n | .. | .. | .. | .. |

data = read.table('tourists.txt', header=T)
head(data)

# Boxplot e grafici
# NOTA: matplot funziona solo se le righe sono elementi indipendenti
boxplot(data, col='gold')
boxplot(scale(x=data, center=T, scale=F), col='gold')     # commenta
pairs(data)                                               # commenta
matplot(t(data), type='l')
boxplot(data, add=T, boxwex=0.1, col = 'red')             # commenta

# -------------------------------------------------------------------------------------
# 2. Mean, Cov, Cor, Var.gen, Var.tot
# -------------------------------------------------------------------------------------
M = sapply(data, mean)
M

S = cov(data)
image(S)
round(S, digits=2)

R = cor(data)
round(R, digits=2)

var.gen = det(S)
var.gen

var.tot =sum(diag(S))
var.tot

# -------------------------------------------------------------------------------------
# 3. PCA on original data (/ based on covariance matrix (S))
# -------------------------------------------------------------------------------------
pc.data = princomp(data, scores=T)
pc.data
summary(pc.data)                        # commenta: look at the cumulative proportion
                                        #           si può fare dimension reduction?

# Proportion of variance explained by each PC
pc.data$sd^2/sum(pc.data$sd^2)

# Cumulative proportion
cumsum(pc.data$sd^2)/sum(pc.data$sd^2)

# Explained variance                    # commenta: dimension reduction?
layout(matrix(c(2,3,1,3),2,byrow=T))
plot(pc.data, las=2, main='Principal components')
barplot(sapply(data,sd)^2, las=2, main='Original Variables', ylab='Variances')
plot(cumsum(pc.data$sd^2)/sum(pc.data$sd^2), type='b', axes=F,
     xlab='number of components', ylab='contribution to the total variance', ylim=c(0,1))
abline(h=1, col='blue')
abline(h=0.8, lty=2, col='blue')
box()
axis(2,at=0:10/10,labels=0:10/10)
axis(1,at=1:ncol(data),labels=1:ncol(data),las=2)
dev.off()

# Loadings
load.data = pc.data$loadings

# Graphical representation of the loadings
# MODIFICARE PAR(MFROW)
par(mfrow=c(2,2))
for( i in 1:dim(data)[2]){
```

```r
  barplot(load.data[,i], ylim=c(-1,1), main=paste("PC",i))
}
dev.off()

# -------------------------------------------------------------------------------------
# 3. PCA on standardized data (/ based on correlation matrix (R))
# -------------------------------------------------------------------------------------
data.sd     = scale(data)
data.sd     = data.frame(data.sd)
pc.data.sd = princomp(data.sd, scores=T)
pc.data.sd
summary(pc.data.sd)

# Proportion of variance explained by each PC
pc.data.sd$sd^2/sum(pc.data.sd$sd^2)

# Cumulative proportion
cumsum(pc.data.sd$sd^2)/sum(pc.data.sd$sd^2)

# Explained variance
layout(matrix(c(2,3,1,3),2,byrow=T))
plot(pc.data.sd, las=2, main='Principal Components', ylim=c(0,7))
abline(h=1, col='blue')
barplot(sapply(data.sd,sd)^2, las=2, main='Original Variables',
        ylim=c(0,7), ylab='Variances')
plot(cumsum(pc.data.sd$sde^2)/sum(pc.data.sd$sde^2), type='b', axes=F,
     xlab='Number of components', ylab='Contribution to the total variance', ylim=c(0,1))
abline(h=1, col='blue')
abline(h=0.8, lty=2, col='blue')
box()
axis(2,at=0:10/10,labels=0:10/10)
axis(1,at=1:ncol(data.sd),labels=1:ncol(data.sd),las=2)
dev.off()

# Loadings
load.data.sd = pc.data.sd$loadings

# Graphical representation of the loadings
# MODIFICARE PAR(MFROW)
par(mfrow=c(4,1))
for( i in 1:dim(data)[2]){
  barplot(load.data.sd[,i], ylim=c(-1,1), main=paste("PC",i))
}

# -------------------------------------------------------------------------------------
# 4. Scores
#    If we want to perform it on the standardized data:
#    data = data.sd
#    pc.data = pc.data.sd
# -------------------------------------------------------------------------------------
scores.data = pc.data$scores
head(scores.data)
summary(scores.data)

# Boxplot original data vs. boxplot PC
layout(matrix(c(1,2),2))
boxplot(data, las=2, col='gold', main='Original data')
scores.data = data.frame(scores.data)
boxplot(scores.data, las=2, col='grey',main='Principal components')
dev.off()

# Plot
plot(scores.data[,1:2])
abline(h=0, v=0, lty=2, col='grey')
text(scores.data[,1], scores.data[,2], dimnames(data)[[1]], cex=0.7)

# Plot
biplot(pc.data)

# -------------------------------------------------------------------------------------
# 5. PROJECTIONS ON THE SPACE OF PRINCIPAL COMPONENTS
#    ATTENZIONE: il dataset deve essere tipo:
#
#          ---------------------------------------------------
#          |         |   day_1   |   day_2   | .. |   day_n   |
#          |         | /feature1 | /feature2 | .. | /feature_n|
#          ---------------------------------------------------
#          |Person_1|     ..     |    ..     | .. |     ..    |
#          |Person_2|     ..     |    ..     | .. |     ..    |
#          ---------------------------------------------------
#    SE E' TRAPOSTO ALLORA:                         data = t(data)
#
#    SE LAVORIAMO CON STANDARDIZZATO ALLORA:        data = data.sd
#                                                   load.data = load.data.sd
# -------------------------------------------------------------------------------------

# Numero delle componenti principali:
numero_pc = dim(data)[2]
```

```r
# 1. "Projection on the space generatebu the k component"
matplot(t(data), type='l', main = 'Data', ylim=range(data))
meanF = colMeans(data)
matplot(meanF, type='l', main = '0 PC', lwd=2, ylim=range(data))
for(i in 1:numero_pc){
  projection <- matrix(meanF, dim(data)[[1]], dim(data)[[2]], byrow=T) +
    scores.data[,i] %*% t(load.data[,i])
  matplot(t(projection), type='l', main = paste(i, 'PC'), ylim=range(data))
  matplot(meanF, type='l', lwd=2, add=T)
}

# 2. "Projection on the space generated by the first k components"
matplot(t(data), type='l', main = 'Data', ylim=range(data))
meanF <- colMeans(data)
matplot(meanF, type='l', main = 'First 0 PCs', lwd=2, ylim=range(data))
projection <- matrix(meanF, dim(data)[[1]], dim(data)[[2]], byrow=T)
for(i in 1:numero_pc){
  projection <- projection + scores.data[,i] %*% t(load.data[,i])
  matplot(t(projection), type='l', main = paste('First', i, 'PCs'), ylim=range(data))
  matplot(meanF, type='l', lwd=2, add=T)
}

# ------------------------------------------------------------------------------------
# 6. SE ci sono etichette (labels)
#    I lables possono essere anche più di uno
#    Plottiamo in 2D
# ------------------------------------------------------------------------------------
head(labels)

# Coloriamo gli scores in base alla prima etichetta (1* colonna)
# consideriamo un nuovo data frame
# scores.data[,1:2] = [,(x,y)]

# Trasformiamo in:
#    | x | y | Livello |
#    -------------------
#    | . | . |   ..    |
#
punti = as.data.frame(scores.data[,1:2])
punti[, 'Livello'] = 0*1:length(punti$Comp.1)
punti[, 'Colore' ] = 0*1:length(punti$Comp.1)

temp    = as.factor(labels[,1])
# temp    = as.numeric(as.factor(labels[,2]))     << Se voglio il secondo label
livelli = levels(as.factor(labels[,1]))
# livelli = levels(as.factor(labels[,2]))         << Se voglio il secondo label
num_lev = length(livelli)
colori  = rainbow(num_lev)

for(i in 1:num_lev){
  ind = which(temp == livelli[i])
  punti$Livello[ind] = i
  punti$Colore[ind]  = colori[i]
}

# Plot
# ATTENZIONE: VA CAMBIATO NEL 'rep' IN BASE A YLIM E XLIM
plot(punti[,1:2], col=punti$Colore, pch=19)
legend('topright', livelli, fill=punti$Colore)
points(punti[,1], rep(-1, length(punti[,1])), col=punti$Colore, pch=19)
points(rep(-2, length(punti[,2])), punti[,2], col=punti$Colore, pch=19)

# ------------------------------------------------------------------------------------
# 7. Calcolare gli scores di un nuovo elemento
#    (in questo caso il data.frame aveva 4 colonne)
# ------------------------------------------------------------------------------------
new_elem        = c(13,10,11,13)
scores.new_elem = t(pc.data$loadings) %*% (new_elem-colMeans(data))
scores.new_elem

# Plot it
plot(scores.data[,1], scores.data[,2], col='gray', xlab='Comp1', ylab='Comp2', pch=19)
points(scores.new_elem[1], scores.new_elem[2], col='red', pch=19)
```

!